인터넷 트래픽 분석을 위한 헤더 시그니쳐 관리 방법

(Header Signature Maintenance for Internet Traffic Identification)

윤성호, 박준상, 김명섭

고려대학교 컴퓨터정보학과

{sungho_yoon, junsang_park, tmskim}@korea.ac.kr

요 약

효과적인 네트워크 관리를 위한 응용 트래픽 분석의 중요성이 강조됨에 따라 다양한 분석 방법론이 제안되었다. 헤더 시그니쳐 기반 응용 분석은 인터넷 응용(서비스)을 제공하는 서버의 3-tuple(IP address, port, L4 protocol)의 헤더 정보만을 이용하기 때문에 기존 방법론의 한계점(암호화, 사생활 침해, 오버헤드 등)을 효과적으로 극복할 수 있지만, 헤더 정보의 조합으로 구성되는 시그니쳐의 특성과 급변하는 네트워크 환경으로 인해 생성된 시그니쳐의 개수가 급격히 증가한다는 문제점을 갖고 있다. 본 논문에서는 수집된 헤더 시그니쳐의 분석 능력을 웨이트로 표현하여 분석 시점에서 최적의 헤더 시그니쳐를 유지하는 효과적인 헤더 시그니쳐 관리 방법을 제안한다. 헤더 시그니쳐의 웨이트를 결정하는 관리 요소는 시그니쳐 사용 기간과 상관성이 큰 특징들로 정의하며, 이를 사용한 관리 함수는 관리 시점의 측정값에 비중을 두어 네트워크 환경 변화에 유연하게 적용할 수 있도록 한다. 또한, 관리 방법의 타당성을 증명하기 위해 다양한 평가 요소를 제안하고 기존 관리 방법과 비교 평가 한다.

Keywords: header signature, network management, traffic classification, traffic identification

1. 서론 [1]

초고속 인터넷의 보급과 인터넷 기반의 서비스가 다양화됨에 따라 네트워크 관리의 중요성이 강조되고 있다[1][2]. 하지만 한정적인 네트워크 자원과 급증하는 트래픽은 네트워크의 부담을 가중시킨다. ISP 나 네트워크 관리자는 망의 안정성과 신뢰성을 확보하기 위해 네트워크 장비의 대역폭을 증가시키고 확장하는 방법을 취하지만, 네트워크 장비의 확충과 성능 향상을 고집하기에는 비용과 기술적인 측면에서 무리가 있다. 따라서 효과적인 네트워크 자원 활용을 위해 응용 레벨 트래픽 분석을 기반으로 사전에 네트워크 소모량이 높은 서비스와 피크 시간대 등을 파악하고 서비스의 사용자 별 이용 패턴을 분석하여 모니터링 해야 한다.

네트워크 트래픽의 응용을 탐지하는 트래픽 분석은 다양한 네트워크 관리 정책들을 적용하기 위해서 반드시 필요한 선행 기술이다. 트래픽 분석 방법론 또는 분석 시스템의 최종목표는 분석하고자 하는 대상 네트워크의 모든 트래픽을 응용 별로 정확하게 분석하는 것이다. 트래픽 분석을 위해 다양한 트래픽 특징을 이용한 방법론들이 제안되었지만, 실제 네트워크 관리에 활용하기에는 많은 한계점을 가지고 있다. 대표적인 한계점으로는 시그니쳐 생성 및 관리의 어려움, 계산 복잡도, 사생활 침해, 실시간 제어 문제 등이 있다.

기존 분석 방법론의 한계점을 극복하기 위하여 본 연구진은 헤더 시그니쳐 기반 응용 트래픽 분석 방법을 제안하였다[3]. 헤더 시그니쳐는 특정 응용(서비스)을 제공하는 서버의 3-tuple(IP address, port, L4 protocol)이다. 헤더 정보를 이용하여 트래픽을 분석하기 때문에 기존 방법론의 한계점들을 효과적으로 극복할 수 있다. 비록, P2P 형태의 트래픽을 발생시키는 응용과 동적 혹은 임의 포트를 사용하는 응용의 출현이 급증하였지만, 서버-클라이언트 형태의 트래픽이 아직까지 많이 사용되고 있고, P2P 응용일지라도

이 논문은 2012 년 정부(교육과학기술부)의 재원으로 한국연구재단(2012R1A1A2007483) 및 2013 년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대정보.컴퓨팅기술개발사업(2010-0020728)의 지원을 받아 수행된 연구임

업데이트 및 로그인 정보는 중앙 서버에서 처리하기 때문에 서버의 헤더 정보는 트래픽 분석에서 큰 의미를 가진다. 실제, 다수의 기존 연구[5][6][7]에서 서버의 헤더 정보를 트래픽 분석에 사용하였다. 기존 연구에서 헤더 정보를 트래픽 분석에 활용하고 관리하는 방법은 2 장 관련 연구에서 자세히 설명한다.

헤더 시그니쳐는 기존 방법론의 한계점을 효과적으로 극복할 수 있지만, 헤더 정보의 조합으로 구성되는 시그니쳐의 특성으로 인해 시그니쳐의 개수가 지속적으로 증가한다는 문제점을 가지고 있다. 실제 시그니쳐로 생성 가능한 조합은 2^{49}(IP:2^{32}, port:2^{16}, protocol:2)이다. 한정적인 물리적 메모리 크기로 인해 모든 시그니쳐를 저장할 수 없을 뿐만 아니라, 시그니쳐 개수의 증가는 분석시스템에 오버헤드로 작용한다. 따라서, 생성된 시그니쳐 중에서 분석에 활용될 가능성이 큰 시그니쳐만을 유지하고 그렇지 않은 시그니쳐는 조기에 삭제하는 효과적인 시그니쳐 관리 방법이 요구된다.

본 논문에서는 효과적인 시그니쳐 관리를 위해 관리 함수에 기반한 관리 방법을 제안한다. 관리 함수는 관리 요소를 기반으로 시그니쳐의 중요도를 의미하는 웨이트를 측정한다. 매 관리 시점(ϵ)마다 측정된 웨이트를 기준으로 더 이상 분석에 활용될 가능성이 적은 시그니쳐(웨이트 0 미만)를 삭제한다. 관리 함수에서 사용하는 관리 요소는 해당 시그니쳐로 분석된 트래픽 통계적 정보(분석한 트래픽의 양, 분석한 트래픽의 클라이언트 개수)로 정의한다. 또한, 관리 함수는 관리 시점의 측정값에 비중을 두어 네트워크 환경 변화에 유연하게 적용할 수 있도록 하였다. 관리 방법의 성능을 측정할 수 있는 다양한 평가 요소를 제안하고 이를 바탕으로 제안된 관리방법의 타당성을 평가한다.

본 논문은 다음과 같은 순서로 기술한다. 2 장에서는 헤더 정보를 이용한 기존 방법론들을 살펴보고, 3 장에서는 헤더 시그니쳐의 정의와 분석 시스템의 구조에 대해 설명한다. 4 장에서는 관리 함수를 제안하고, 5 장에서는 제안한 관리 함수의 성능을 평가한다. 마지막으로 6 장에서는 결론과 향후 연구를 언급한다.

2. 관련연구

트래픽을 분석하는 가장 원시적인 방법은 포트 번호를 이용한 방법이다. IANA[4]에서 정의한 포트 번호를 이용하여 트래픽을 분석한다. 하지만 방화벽을 원활히 통과하기 위해 동적 또는 임의 포트 번호를 사용하는 응용이 출현하고 있는 상황에서 포트 기반 분석 방법은 70% 이하의 정확도를 가진다[5]. 포트 기반 분석 방법의 낮은 정확도를 보완하기 위해 헤더 정보를 이용한 다양한 트래픽 분석 방법이 제안되었다. 이러한 방법들의 공통적인 가정은 인터넷 응용(서비스) 서버는 일정 기간 동안 지속적으로 동일한 응용을 제공한다는 것이다.

트래픽의 다른 특징들(페이로드, 통계정보 등)을 사용하는 방법론들에 비해 헤더 정보를 이용한 트래픽 분석 방법은 많은 이점을 가지고 있다. 대표적인 장점은 다음과 같다.

- 트래픽 수집 시 발생할 수 있는 패킷 손실(loss)과 단편화(fragmentation), 샘플링(sampling)으로 인한 분석 성능 저하가 전혀 없다.
- 헤더 정보만을 분석하기 때문에 페이로드 암호화로 인한 분석 성능 저하의 문제가 발생하지 않는다.
- 트래픽 분석 시 시그니쳐와 트래픽의 고정된 위치에 존재하는 헤더 정보를 비교하기 때문에 매우 빠르게 분석할 수 있다.

Moore 등[5]은 포트 기반 분석 방법의 부 정확성을 다양한 실험을 통해 증명하였다. 이 논문은 {host/port} 헤더 정보를 특수한 종류의 트래픽(포트 스캐닝, 음성 스트리밍 등)을 검증하기 위해 사용하였다. 수집된 헤더 정보의 관리를 위하여 해당 호스트가 트래픽을 발생하는 시점에만 유지하는 단순한 관리 방법을 사용하였다.

Karagiannis 등[6]은 P2P 응용 트래픽을 분석하기 위해 트래픽 발생 형태와 통계적 특성을 사용하였다. 이를 위해 {TCP/UDP IP pairs}, {IP,port}를 P2PIP 리스트로 구성하고 분석에 활용하였다. 이 논문은 제안하는 방법론에 초점을 맞추었기 때문에 P2PIP 리스트의 관리에 대해서는 언급하지 않고 계속 누적하는 원시적인 방법을 취한다.

Baldi 등[7]은 응용 트래픽 분석을 위하여 “network coordinates” (IP address, TCP/UDP port)를 기존 분석 방법(페이로드) 결과에서 추출하여 서비스 테이블을 구성하고 이를 응용 분석에 활용하였다. 서비스 테이블의 각 원소들은 일정 시간이상 사용되지 않으면 삭제되는 단순한 관리 방법이 적용하였다.

본 연구진의 기존 연구[3]에서는 서버의 3-tuple(IP address, port, L4 protocol)을 헤더 시그니쳐로 정의하고 분석된 트래픽의 특성과 시그니쳐 사용이력을 기반으로 시그니쳐를 관리하는 방법을 제안하였다. 즉, 시그니쳐 특성을 abnormal, timeout, dominant, popular 로 구분하고 특정 조건에 만족하면 삭제하는

방법을 사용하였다. 하지만, 시그니쳐의 특성을 파악하기 위한 과정이 복잡하고, 등록 이후의 모든 측정값들을 누적 적용하여 네트워크 환경 변화에 유연하게 적용되지 않는 문제점이 있었다.

기존의 다양한 논문을 통해 응용을 제공하는 서버의 헤더 정보가 트래픽 분석에 효과적으로 활용될 수 있음을 확인할 수 있다. 하지만, 헤더 정보의 사용 방법에만 초점을 맞추었고, 원시적이거나 매우 간단한 관리 방법만을 제시하였다. 특히, 일정 시간 이상 사용하지 않는 헤더 정보에 한해 삭제하는 방법은 응용의 사용 주기를 고려하지 않아 상대적으로 사용 주기가 긴 응용의 헤더 정보가 빈번히 삭제되는 문제점을 야기시킨다. 비록, 기존 연구에서 시그니쳐의 특성을 파악하여 관리하는 방법을 제안하였으나, 관리 방법이 복잡하고 네트워크 환경 변화에 유연하게 적용되지 않는 문제점이 발생하였다. 따라서 헤더 정보, 즉 헤더 시그니쳐의 특성을 파악하여 분석에 활용될 가능성이 높은 시그니쳐를 오래 동안 유지시키고 그렇지 않은 시그니쳐를 조기에 삭제하는 보다 효율적인 관리 방법이 요구된다.

3. 헤더 시그니쳐

본 장에서는 헤더 시그니쳐를 정의하고 분석 시스템의 구조에 대해 설명한다. 헤더 시그니쳐는 특정 응용(서비스)을 제공하는 서버의 3-tuple(IP address, port, L4 protocol)과 응용(서비스), 그리고 관리를 위한 웨이트로 구성된다.

트래픽의 발생 형태가 복잡해짐에 따라 플로우 단위가 제안되었다. 플로우는 5-tuple(source IP address, source port, destination IP address, destination port, L4 protocol)이 동일한 패킷들을 하나의 단위로 모은 것이다. 수식 (1)과 같이 F 는 플로우(f)의 집합으로, F(X=x)는 플로우의 속성 X 가 특정 x 값을 가지는 플로우의 집합으로 정의한다. 즉, 수식 (2)와 같이 F(dstIP=dip, dstPort=dport)는 F 에 속한 플로우 중 destination IP address 가 dip 이고 destionation port 가 dport 인 모든 플로우들의 집합을 의미한다.

$$F = \{f_1, f_2, f_3, \cdots, f_n\} \tag{1}$$

$$F(X = x) = \{f \mid f \in F, f(X) = x,\ X : 5 - tuple\ \} \tag{2}$$

플로우 단위는 패킷 단위에 비해 대용량 트래픽 분석에 효과적이지만, 특정한 두 호스트의 포트(end point)를 기준으로 트래픽을 구분하기 때문에 발생 형태에 대한 충분한 정보를 제공하지 않는다. 특히, 서버-클라이언트 형태의 트래픽에서 플로우 간의 관계를 파악하는 것은 추가적인 분석을 요구한다.

$$B = \{b \mid b = \{f \mid f \in F(x), x = \{IP, Port, Prot\}\}\} \tag{3}$$

효과적인 시그니쳐 생성을 위하여 플로우 간 발생 형태를 반영한 트래픽 단위 “Bunch”를 정의한다. 수식 (3)과 같이 Bunch 는 3-tuple(IP address, port, L4 protocol)가 동일한 플로우들의 집합이다. 즉, 서버의 특정 포트에서 발생되는 모든 플로우들을 의미한다. 서버 노드와 1 개 이상의 상대 호스트들 사이에서 발생한 모든 플로우들의 집합이다. TCP 의 경우 서버를 기준으로 bunch 를 생성하고, UDP 의 경우는 서버 클라이언트의 구분이 없으므로 양쪽 end-host 모두를 기준으로 bunch 를 생성한다.

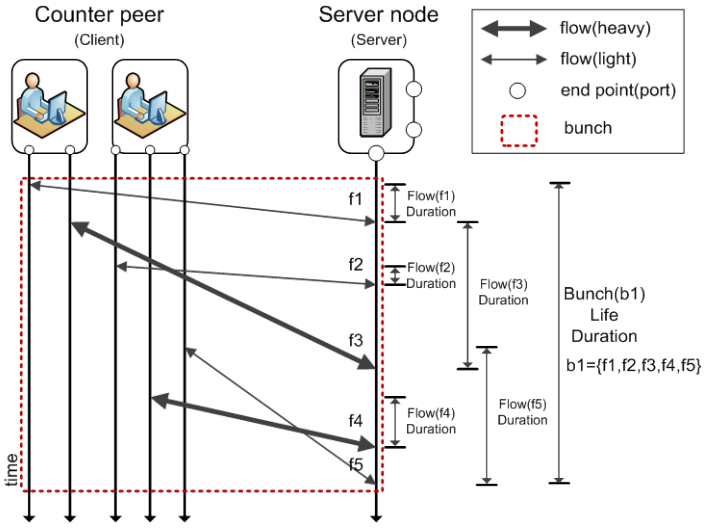

그림 1. Structure of bunch

그림 1 은 Bunch 의 구조에 대해 나타낸다. 응용을 제공하는 서버와 2 대의 호스트(클라이어트) 사이에서 발생하는 트래픽을 예로 들어 Bunch 의 구조를 설명한다. 양방향 화살표 선은 서버의 특정 포트와 클라이언트의 여러 포트 사이에서 발생한 플로우를 의미한다. 선의 굵기는 플로우의 통계적 특성(packet, byte)을 반영하여 heavy 플로우인 경우 굵게, light 플로우인 경우 얇게 표시하였다. 또한, 화살표 양끝 지점의 위치는 플로우의 시작 및 종료 시간을 의미한다. Bunch 는 특정 서버 포트를 중심으로 발생한 모든 플로우들의 집합이기 때문에 빨간 박스 안에 포함된 모든 플로우들은 하나의 Bunch 로 재 구성된다.

본 논문에서 제안하는 헤더 시그니쳐는 수식 (4), (5)와 같이 Bunch 로 재구성된 트래픽의 서버 노드를 의미한다. 헤더 시그니쳐는 헤더 정보(x), 응용 이름(a), 웨이트(w)로 구성된다. 헤더 정보는 서버 노드의 3-tuple(IP address, port, L4 protocol)이며, 응용은 해당 서버 노드가 제공하는 응용(서비스)의 이름을 의미한다. 웨이트는 시그니쳐 관리를 위해 해당 시그니쳐로 분석된 트래픽의 통계적 특징을 이용하여 계산된 값이다. 즉, 관리 함수(M(hs))를 통해 계산된 값을 의미한다. 관리 함수에 대한 설명은 4 장에서 자세히 설명한다.

$$HS = \{hs_1, hs_2, hs_3, \ldots, hs_m\} \tag{4}$$

$$hs = \{(x, a, w) | x = \{IP, port, prot\}, w = M(hs)\} \tag{5}$$

Feature	Explanation
FT(First Time)	First time used to identify traffic
LT(Last Time)	Last time used to identify traffic
LD(Life Duration)	Duration of identification (LT- FT)
FC(Flow Count)	Flow count of identified(extracted) traffic
PC(Packet Count)	Packet count of identified(extacted) traffic
BC(Byte Count)	Byte count of identified(extracted) traffic
CPC(Counter Peer Count)	Number of client host in identified(extracted) traffic

표 1. Features of header signature

표 1 은 헤더 시그니쳐의 다양한 특징을 나타낸다. FT 는 해당 시그니쳐가 분석을 시작한 시점이고 LT 는 마지막으로 트래픽을 분석한 시점이다. LD 는 FT 와 LT 의 차이이며, 시그니쳐가 분석에 사용된 기간을 의미한다. FC, PC, BC 는 각각 분석 또는 시그니쳐 생성에 사용된 트래픽의 통계 정보를 의미하며, CPC 는 분석 또는 시그니쳐 생성에 사용된 트래픽의 클라이언트 호스트 개수를 의미한다. 이러한 시그니쳐 특징을 관리 요소로 정의하며 이를 사용한 관리 함수에 의해 시그니쳐의 웨이트 값이 계산된다.

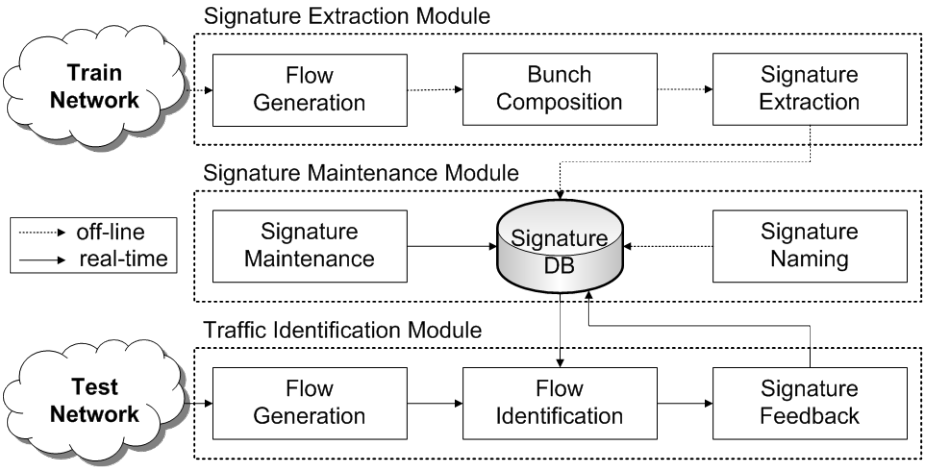

그림 2. System architecture of identification system

그림 2 는 헤더 시그니쳐 기반 분석 시스템의 구조를 나타낸다. 분석 시스템은 크게 3 부분으로 구성된다. 생성 네트워크에서 시그니쳐를 생성하는 시그니쳐 생성부, 생성된 시그니쳐를 적용하여 트래픽을 분석하는 트래픽 분석부, 그리고 시그니쳐의 응용을 명명하고 관리하는 시그니쳐 관리부이다.

본 논문에서 제안하는 분석 시스템은 좀 더 풍부한 시그니쳐를 얻기 위해 시그니쳐의 생성과 응용(A)을 명명하는 부분을 분리하였다. 응용을 알고 있는 트래픽(Ground-Truth Traffic)만을 대상으로 시그니쳐를 생성할 경우 시그니쳐의 추출 범위가 줄어들기 때문에 생성 네트워크의 모든 트래픽을 대상으로 시그니쳐를 생성한다. 즉, 생성 네트워크의 모든 트래픽을 대상으로 시그니쳐의 헤더정보(X)를 생성하고 생성부 또는 관리부에서 다양한 분석 방법(페이로드, 통계 등)의 결과를 사용하여 응용(A)을 명명함으로써 좀 더 많은 시그니쳐를 생성하여 트래픽 분석 성능을 향상시킨다.

Input : $T_{train} = \{t_1, t_2, t_3, \dots, t_a\}$, $HS_{old} = \{hs_1, hs_2, hs_3, \dots, hs_n\}$
Output : $F = \{f_1, f_2, f_3, \dots, f_b\}$, $B = \{b_1, b_2, b_3, \dots, b_c\}$,
$HS_{new} = \{hs_1, hs_2, hs_3, \dots, hs_m\}$

```
Signature-Extraction(T)
1:  for i=1 to a                                   //Flow Generation
2:      F=F∪makeFlow(t_i)
3:  for i=1 to b                                   //Bunch Composition
4:      B=B∪makeBunch(f_i)
5:  for i=1 to c                                   //Signature Extraction
6:      hs_temp=getServerNode(b_i)
7:      hs_temp.setWeight(M())
8:      hs_temp.setApplication() if possible
9:      If hs_temp is new signature
10:         HS=HS∪ hs_temp
11:     Else
12:         updateAlreadySignature(hs_temp)
13: return HS
```

Algorithm 1. Header signature extraction

알고리즘 1 은 시그니쳐 생성부의 수행 과정을 나타낸다. 시그니쳐 생성부에서는 생성 네트워크에서 수집한 트래픽을 플로우 단위로 변경(Flow Generation)하고 Bunch 단위로 재조합(Bunch Composition)한다. 시그니쳐는 Bunch 의 서버 노드에서 3-tuple 을 추출하여 생성(Header Signature Extraction)한다. 생성된 시그니쳐는 4 장에서 설명할 관리 함수를 기반으로 초기 웨이트 값(w)을 설정하고 해당 시그니쳐의 응용 이름(a)이 다양한 분석 방법으로 판별된 경우 해당 응용을 명명한다. 만약 응용 이름을 모르는 경우에는 추후 관리부에서 명명한다. 생성이 완료된 시그니쳐는 시그니쳐 DB 에 동일한 시그니쳐가 존재하지 않는 경우 추가하고 이미 존재하는 경우 웨이트 값을 갱신한다.

Input: $T_{test} = \{t_1, t_2, t_3, \dots, t_a\}$, $HS = \{hs_1, hs_2, hs_3, \dots, hs_n\}$
Output : $F_{identified} = \{f_1, f_2, f_3, \dots, f_b\}$

```
Signature-Identification(T, HS)
1   for i=1 to a                                   //Flow Generation
2       F=F∪makeFlow(t_i)
3   for i=1 to b                                   //Flow Identification
4       {IP,port,prot}=getServer(f_i)
5     for j=1 to n
6         if both{IP,port,prot} and hs_i({IP,port,prot}) are same
7             F=F∪f_i.setApplication(hs_i(a))
8         hs_j.updateElement(f_i)        //Signature Feedback
9   return F
```

Algorithm 2. Traffic identification

알고리즘 2 는 시그니쳐 분석부의 수행 과정을 나타낸다. 트래픽 분석부는 분석 대상이 되는 분석 네트워크의 트래픽을 플로우 단위로 변경(Flow Generation)하고 시그니쳐 DB 에 저장된 시그니쳐와 헤더 정보 비교를 통해 트래픽을 분석(Flow Identification)한다. 즉, 동일한 헤더 정보(X)를 가지는 경우 해당 시그니쳐의 응용(A)을 트래픽에 기록한다. 또한, 시그니쳐 관리를 위해 분석된 트래픽을 기반으로 해당 시그니쳐의 관리 요소 값을 갱신(Signature Feedback)한다.

Input: $HS_{old} = \{hs_1, hs_2, hs_3, \ldots, hs_o\}$
Output : $HS_{new} = \{hs_1, hs_2, hs_3, \ldots, hs_n\}$
Signature-Maintenance(HS)

```
1   for i=1 to o                                  //Signature Maintenance
2       hs_i.setWeight(M())
3       if hs_i(w) is under 0
4           delete hs_i from signature DB
5   for i=1 to n                                  //Signature Naming
6       hs_i.setApplication() if possible
7   return HS
```

Algorithm 3. Header signature maintenance

알고리즘 3 은 시그니쳐 관리부의 수행 과정을 나타낸다. 시그니쳐 관리부에서는 갱신된 관리 요소를 관리 함수에 적용하여 시그니쳐 웨이트를 산출하고 이를 기준으로 불필요한 시그니쳐를 삭제(Signature Maintenance)한다. 또한, 다양한 분석 방법(페이로드, 통계, 에이젼트 기반 등)을 통해 시그니쳐의 응용(A)을 시그니쳐에 명명(Signature Naming)한다.

4. 시그니쳐 관리

시그니쳐 관리의 최종 목적은 분석에 활용될 가능성이 큰 최적의 시그니쳐를 장기간 시그니쳐 DB 에 유지시키는 것이다. 즉, 최소한의 시그니쳐로 최대의 분석 성능(분석률, 정확도, 분석시간)을 얻는 것을 목표로 한다.

Name	Date	Duration	Local IP	Remote IP	Flow(K)	Packet(M)	Byte(G)	Avg. Mbps
KU-Cam-01	2012.03.01	31days	10,494	47,040,918	659,694	53,906	47,848	142,91
KU-Cam-02	2012.06.01	30days	11,437	43,440,289	774,448	59,238	50,704	156.49
KU-Cam-03	2012.10.01	92days	195	35,924,910	295,071	20,673	16,182	16.28

표 2. Traffic Trace

표 2 는 헤더 시그니쳐의 특성과 제안한 관리 방법의 타당성을 검증하기 위한 실험에 사용한 트래픽 정보를 나타낸다. 트래픽은 학내 망에서 학기초(KU-Cam-01), 학기말(KU-Cam-02), 학기전체(KU-Cam-03)로 나누어 다양한 기간 동안 수집하였다.

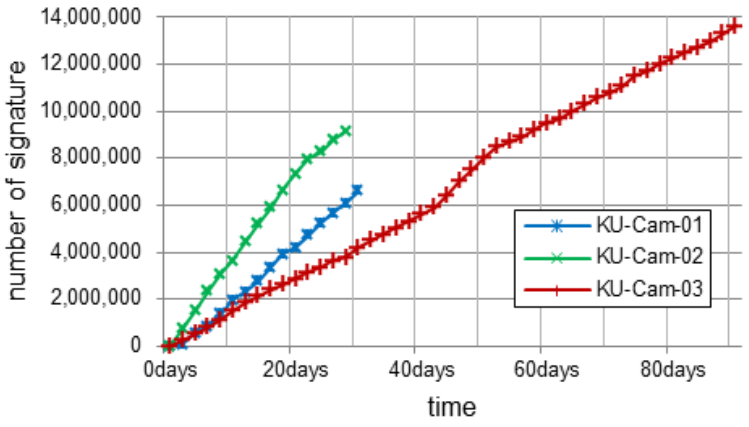

그림 3. Change of signature count

그림 3 은 표 2 에 제시한 트래픽을 관리 모듈을 적용하지 않고 (accumulated method) 생성 시스템에 적용하였을 때, 생성되는 시그니쳐의 개수 추이를 보여준다. 즉, 생성된 시그니쳐를 삭제하지 않고 시그니쳐 DB 에 저장하였을 때, 증가하는 시그니쳐 개수의 추이를 나타낸다. 그림 3 에서와 같이 시그니쳐 관리 방법을 적용하지 않을 경우 지속적으로 시그니쳐 개수가 증가함을 확인할 수 있다.

시그니쳐를 저장하는 DB 의 용량은 제한되어 있기 때문에 모든 시그니쳐를 저장하는 것은 불가능하다. 따라서, 생성된 시그니쳐 중에서 분석에 활용될 가능성이 큰 시그니쳐는 DB 에 유지하고, 그렇지 않은 시그니쳐는 삭제하여야 한다. 하지만, 시그니쳐의 관리는 실시간으로 적용되어야 하기 때문에 해당 시그니쳐가 향후 분석에 활용될지 아닐지 예측하는 것은 쉽지 않은 작업이다. 따라서 본 장에서는 시그니쳐의 특징 중 LD 와 상관 관계가 있는 특징들을 살펴보고 이를 관리요소로 정의한다. 또한, 정의된 관리 요소를 사용한 관리함수를 제안한다.

관리 요소를 정의하기 위해 한 달간 학내망에서 발생한 트래픽을 수집하여 시그니쳐를 생성하였다(KU-Cam-01). 생성된 시그니쳐의 특성을 파악하기 위해 분석된 트래픽의 통계 정보를 기반으로 시그니쳐의 특징 값을 갱신하였다.

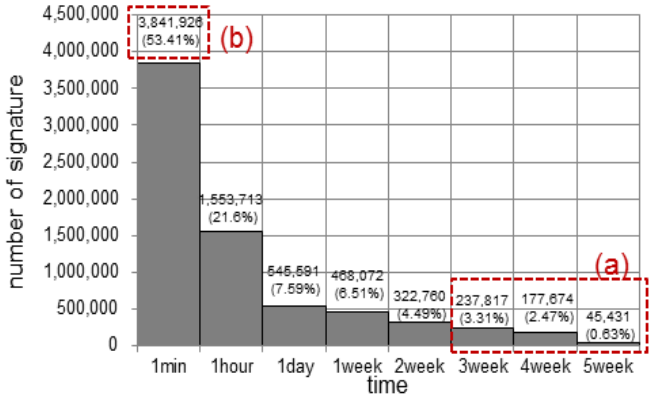

그림 5. Histogram of signature LD(Life Duration)

그림 5 는 생성된 시그니쳐의 LD 를 측정하여 히스토그램으로 나타낸 것이다. x 축은 은 시그니쳐 LD 를 의미하며 y 축은 해당 LD 를 가지는 시그니쳐 개수를 의미한다. 각 막대 상단에 해당 LD 를 가지는 시그니쳐의 개수와 전체에 대한 비율을 표시하였다.

LD 는 시그니쳐가 트래픽을 분석한 첫 시점과 마지막 시점 간의 기간을 의미한다. 큰 LD(2 주이상)를 가지는 시그니쳐가 전체의 6.41%(a)인데 반해, 전체 시그니쳐의 53.41%(b)는 최소 측정 단위인 1 분 미만의 LD 를 보였다. 즉, 과반 이상의 시그니쳐는 생성 시에만 트래픽 분석에 사용되고 더 이상 사용되지 않음을 알 수 있다. 본 장에서는 1 분 미만의 LD 를 가지는 시그니쳐를 “ flash signature”로 정의하고 특징을 분석한다.

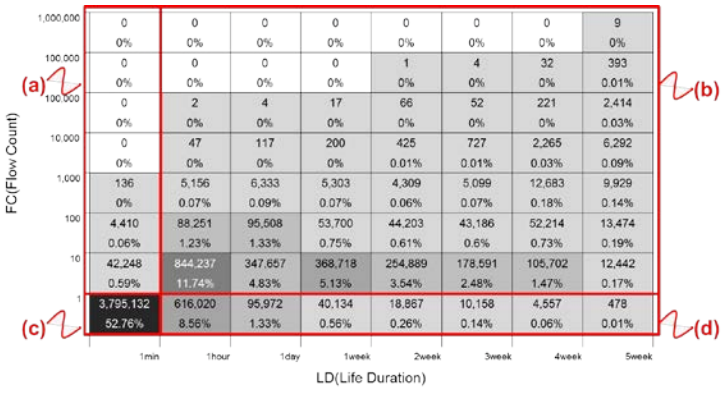

그림 6. Correlation between LD(Life Duration) and FC(Flow Count) of signature

그림 6 은 헤더 시그니쳐의 LD 와 FC 상관관계를 보여준다. x 축은 시그니쳐 LD 를 나타내며, y 축은 시그니쳐 FC 를 나타낸다. 각 박스에 기록된 수치는 해당 구간에 속한 시그니쳐의 개수와 전체에 대한 비율이다. 예를 들어 시그니쳐 LD 가 1 분 이하이고 FC 가 1 개이하인 시그니쳐는 총 3,795,162 개 이며 전체 비율로 52.76%를 차지한다.

분석 기간이 1 분 미만인 flash signature(a+c)의 98.78%(c/(a+c))는 등록 이후 오직 1 개의 플로우만을 분석하였다. 이에 반해, 분석 기간이 1 분 이상인 non-flash signature(b+d)의 경우, 23.46%(d/(b+d))만이 1 개의 플로우를 분석하였다. 즉, 분석 기간이 짧을수록 분석하는 트래픽의 양이 적은 것을 확인 할 수 있다. 지면이 한정적인 관계로 LD 와 BC 의 상관관계를 표현하지 못하지만, FC 와 마찬가지로 비슷한 상관관계를 보였다

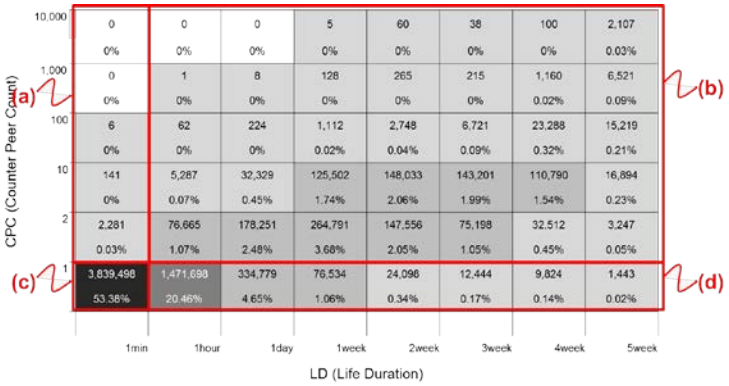

그림 7. Correlation between LD(Life Duration) and CPC(Counter Peer Count) of signature

그림 7 은 헤더 시그니쳐의 LD 와 CPC 상관관계를 보여준다. x 축은 시그니쳐 LD 를 나타내며, y 축은 시그니쳐 CPC 를 나타낸다. 그림의 각 수치는 그림 6 과 같은 의미를 가진다.

flash signature(a+c)의 99.93%(c/(a+c))는 오직 1 개의 클라이언트에서 발생한 트래픽을 분석하였다. 이에 반해, non-flash signature(b+d)의 경우 57.61%(d/(b+d))만이 1 개의 호스트와 트래픽을 발생 하였다. 즉, flash signature 는 특정 호스트에서만 발생하는 트래픽을 분석한다는 것을 확인할 수 있다.

Name	Flash Signature	Non-Flash Signature	Total
Num. of Signatuer	3,841,926 (53.41%)	3,351,058 (46.58%)	7,192,984 (100%)
Identified Flows	3,354,672 (1.26%)	261,761,224 (98.73%)	265,115,896 (100%)
Identified Packets(K)	332,457 (1.00%)	32,786,180 (98.99%)	33,118,637 (100%)
Identified Bytes(M)	282,336 (0.96%)	29,093,558 (99.03%)	29,375,895 (100%)

표 3. Comparison of flash signature and non-flash signature

표 3 은 분석 기간이 1 분 미만인 flash signature 와 1 분 이상인 non-flash signature 의 개수와 해당 시그니쳐들이 분석한 트래픽의 양을 나타낸다. 생성된 시그니쳐의 53.41%인 flash signature 는 전체 트래픽 대비 flow 단위 1.26%, byte 단위 0.96%에 해당하는 트래픽만을 분석하였다. 위의 실험 결과를 종합해 보면 시그니쳐의 대다수를 차지하는 flash signature (LD 값이 1 분 미만)은 생성된 시그니쳐의 과반을 차지하지만, 실제 분석에서는 적은 양의 트래픽만을 분석하였다. 또한, 특정 클라이언트에서 발생하는 트래픽을 분석함으로써, 응용을 대표하는 시그니쳐의 기능을 하지 못한다는 것을 확인할 수 있다.

Feature	Correlation coefficient
FC(Flow Count)	0.027
PC(Packet Count)	0.005
BC(Byte Count)	0.007
CPC(Counter Peer Count)	0.684

표 4. Correlation coefficient of signature feature compare to LD

$$\gamma = \frac{\sum(xy)}{\sqrt{\sum x^2}\sqrt{\sum y^2}} \quad (6)$$

시그니쳐 LD 에 영향을 미치는 시그니쳐 특징을 분석하기 위해 상관관계 분석을 하였다. 상관관계 분석은 수식 6 과 같이 Pearson 상관계수를 이용하였다. 표 4 와 같이 모든 특징들이 시그니쳐 LD 와 상관성을 보였다. 특히 시그니쳐 FC 와 CPC 는 다른 특징들에 비해 큰 상관 관계를 보이는 것을 확인할 수 있다.

위 실험을 기반으로 관리 함수에서 사용할 관리 요소를 정의하였다. 본 논문에서 제안하는 관리 함수에 사용할 관리 요소는 시그니쳐 특징 중 CFC, FC, BC 이다. hs(CFC)는 해당 시그니쳐가 분석 트래픽을 발생시킨 클라이언트의 개수를 의미하고, hs(FC), hs(BC)는 각각 분석된 트래픽의 플로우 개수, 바이트 개수를 의미한다.

$$M_{proposed}(hs) = M_t(hs) + M_{t-1}(hs) - \mu \quad (7)$$

$$M_t(hs) = (\alpha \cdot R_t(CFC)) + (\beta \cdot R_t(FC)) + (\gamma \cdot R_t(BC)) \quad (8)$$

$$R_t(x) = \frac{min(hs_t(x), th_x)}{\min(HS_{max,t}(x), th_x)} \quad (9)$$

본 논문에서 제안하는 관리 함수는 수식 7~9 과 같이 누적된 시그니쳐 특징 값을 사용하는 것이 아니라 매 관리 주기(ϵ)마다 새로 측정된 값과 이전 값을 더하여 계산한다. 또한, 이전 값이 계속 유지되는 것을 방지하기 위하여 매 관리 주기 마다 μ 를 감소시킴으로써 더 이상 트래픽을 분석하지 않는 시그니쳐의 웨이트값을 감소시킨다. 예를 들어 관리 주기가 1 분이고 μ의 값이 1 인 경우, 매분 계산된 웨이트 값에서 1 씩 감소시킨다. 웨이트 값이 0 미만이 되면 시그니쳐 DB 에서 해당 시그니쳐를 삭제한다. 특정 시점(t)의 웨이트는 앞서 정의한 관리 요소를 사용한다. $HS_{max,t}$(CFC)의 의미는 모든 헤더 시그니쳐의 CFC 값 중에서 최대 값을 의미하며, hs_t(CFC)는 해당 시그니쳐의 CFC 값을 의미한다. 따라서 DB 에 존재하는 시그니쳐 중 분석에 가장 많이 활용된 시그니쳐와 관리 대상 시그니쳐의 특징 값 비율을 이용하여 웨이트를 계산한다. 또한, 특징값들의 임계값을 두어 적절한 웨이트값이 계산되도록 한다. 예를 들어 위 관리 함수를 시그니쳐에 적용하였을 경우, 0 과 $\alpha+\beta+\gamma$ 사이의 웨이트 값이 계산된다. 본 논문에서는 관리 주기(ϵ)를 1 분으로 설정하고 μ의 값을 1 로 설정하였다. 즉, 매분 변경된 시그니쳐의 관리 요소를 기반으로 웨이트를 계산하며, 이전 분에 계산된 웨이트 값은 1 씩 감소한다.

5. 실험 및 결과

본 논문에서 제안한 관리 함수의 타당성을 보이기 위하여 다양한 평가 요소를 정의하고 한 달간 트래픽(KU-Cam-01)을 대상으로 분석 시스템을 적용하였다. 관리 함수의 성능을 평가하기 위하여 시그니쳐 관리부의 응용 명명 모듈은 적용하지 않았다. 또한, 객관적인 평가를 위해 본 논문에서 제안하는 관리 방법론(proposed method), 시그니쳐 관리를 적용하지 않고 생성된 모든 시그니쳐를 누적시키는 방법론(accumulated method), 그리고 수식 (10)과 같이 단순히 일정 시간 (α) 사용되지 않은 시그니쳐를 삭제하는 방법론(timeout method)[7]을 비교 실험하였다. Proposed method 와 timeout method 에 사용한 임계값은 1, 60, 1440 으로 다양하게 설정하였다.

$$M_{timeout}(hs) = \alpha - (current\ time - hs(LT)) \quad (10)$$

실험 결과의 객관적 평가를 위하여 수식 (11)~(15)와 같이 분석률, 시그니쳐 개수, 시그니쳐 LD 의 평균, 시그니쳐의 사용률, 그리고 관리 값(시그니쳐 DB 에 등록 및 삭제의 횟수)을 평가 요소로 정의하였다. 분석률은 한 달 트래픽 전체를 대상으로 측정하였고, 나머지 평가 요소는 매 관리 시점(1 분)마다 측정하여, 평균, 최대, 그리고 실험이 종료되는 시점의 값을 계산하였다.

$$\text{Completeness} = \frac{\text{Identified Traffic}}{\text{Total Traffic}} \quad (11)$$

$$Number\ of\ Signature = n(HS) \quad (12)$$

$$Average\ LD = \frac{\sum hs(LD)}{n(HS)} \quad (13)$$

$$Usage\ Rate = \frac{used\ hs}{n(HS)} \quad (14)$$

Maintenance Method		Completeness		Num. of Signature			Average LD (Life Duration)			Usage Rate		
		Flow (%)	Byte (%)	Avg. (K)	Max. (K)	Final (K)	Avg. (sec)	Max. (sec)	Final (sec)	Avg. (%)	Max. (%)	Final (%)
accumulated method		79.45	95.64	18,159	35,399	35,399	24,827	45,609	45,609	0.03	99.00	0.01
Proposed method	α, β, γ 1	39.93	86.15	3	35	1	57,531	1,654,796	157,983	60.31	99.00	64.00
	α, β, γ 60	62.34	90.92	119	1,290	53	150,904	771,569	443,865	1.88	99.00	2.00
	α, β, γ 1440	67.57	93.03	2,308	4,833	1,445	136,716	377,597	356,613	0.01	99.00	0.01
Timeout method	α 1	39.88	85.73	3	35	1	6,161	97,576	22,575	60.27	100	67.00
	α 60	59.58	89.41	60	853	23	4,759	48,134	8,996	3.89	99.00	4.00
	α 1440	66.45	91.90	1,104	3,719	477	12,231	40,769	25,266	0.03	99.00	0.01

표 5. Experimental result

표 5 의 실험 결과와 같이 분석률 측면에서는 accumulated method 가 가장 좋은 성능을 보인다. 시그니쳐의 삭제 없이 생성된 시그니쳐를 계속 누적하여 분석에 사용하기 때문에 가장 높은 분석률을 보인다. 하지만, proposed method 와 timeout method 의 임계값을 1440 으로 설정할 경우, accumulated method 와 비등한 분석률을 보인다. 즉, 생성된 시그니쳐를 하루동안(1440 분)만 유지하더라도, 모든 시그니쳐를 유지하는 것과 비슷한 성능을 보이는 것이다.

시그니쳐 개수를 비교해 보면 임계값을 1440 으로 설정하고 분석이 종료된 시점(final)을 기준으로 accumulated method 는 proposed method 보다 약 24 배 정도 큰 것을 확인 할 수 있다. 즉, 비등한 분석 성능에 비해 보유한 시그니쳐 개수가 훨씬 많음을 확인할 수 있다. 이는 시그니쳐 보유를 위한 물리적 메모리와 분석 시간에 큰 오버헤드를 요구한다.

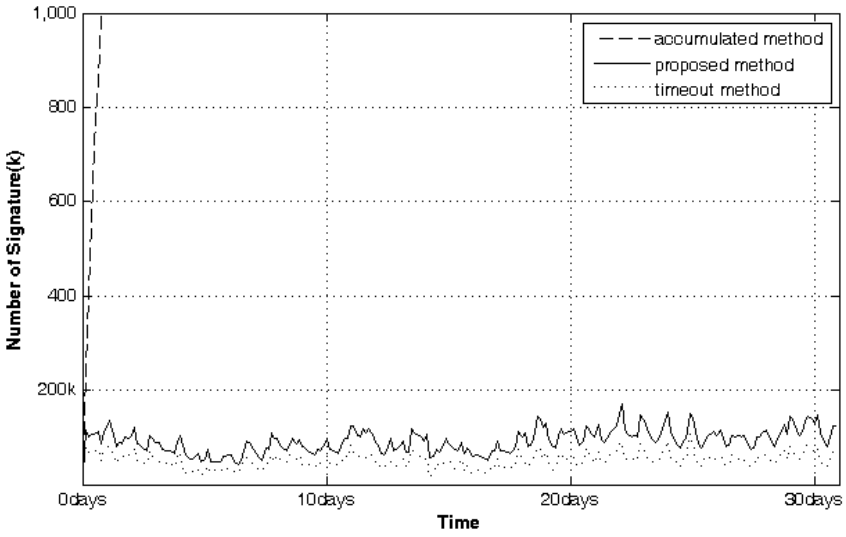

그림 8. Comparison of number of signature (y축 log scal)

그림 8 은 시험 기간 동안 변화하는 시그니쳐 개수를 나타낸다. accumulated method 는 시그니쳐의 개수가 급격히 증가하고 proposed method 와 timeout method 는 일정 수준을 유지하는 것을 확인 할 수 있다.

시그니쳐 LD 의 평균 측면에서는 proposed method 가 독보적으로 좋은 성능을 가진다. 임계값 1440, final 시점을 기준으로 accumulated method 에 비해 약 7 배, timeout method 에 비해 약 14 배 큰 것을 확인할 수 있다. 시그니쳐 LD 의 평균이 크다는 의미는 시그니쳐 DB 에 보유한 시그니쳐가 오랜 기간 동안 트래픽 분석에 사용되었다는 의미이다. 즉, 분석에 사용되는 기간이 긴 시그니쳐를 잘 유지하여 분석에 활용한다는 의미이다. 시그니쳐 사용율과 관리값은 세 관리 방법 모두 비슷한 수치를 보였다.

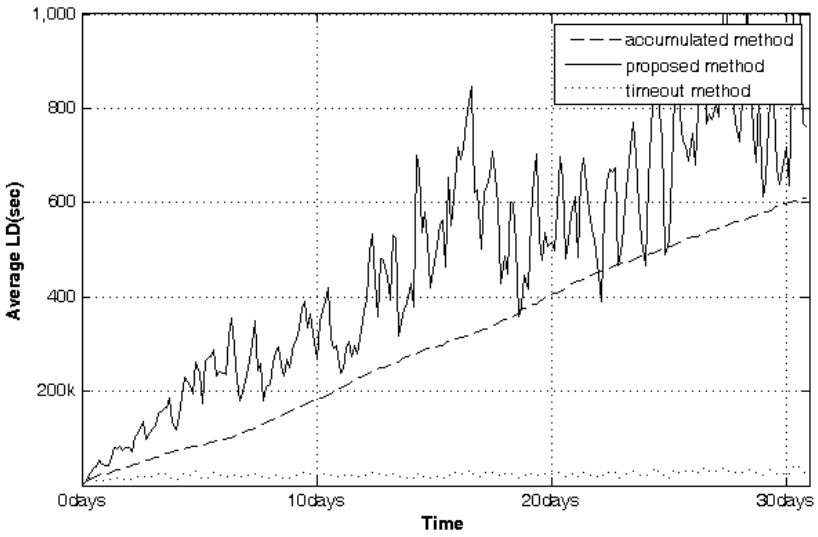

그림 9. Comparison of average LD

그림 9 는 매 관리 시점에 측정된 시그니쳐 평균 LD 의 변화를 비교한 것이다. timeout method 의 경우 시그니쳐의 삭제가 빈번히 이루어 지기 때문에 낮은 평균 LD 를 유지하는 것을 확인할 수 있다. 이에 반해 proposed method 는 다른 두 관리 방법 보다 높은 평균 LD 를 유지한다. 즉, 단기간 사용하는 flash signature 를 조기에 삭제하고 그렇지 않은 시그니쳐를 잘 유지하는 것을 확인 할 수 있다.

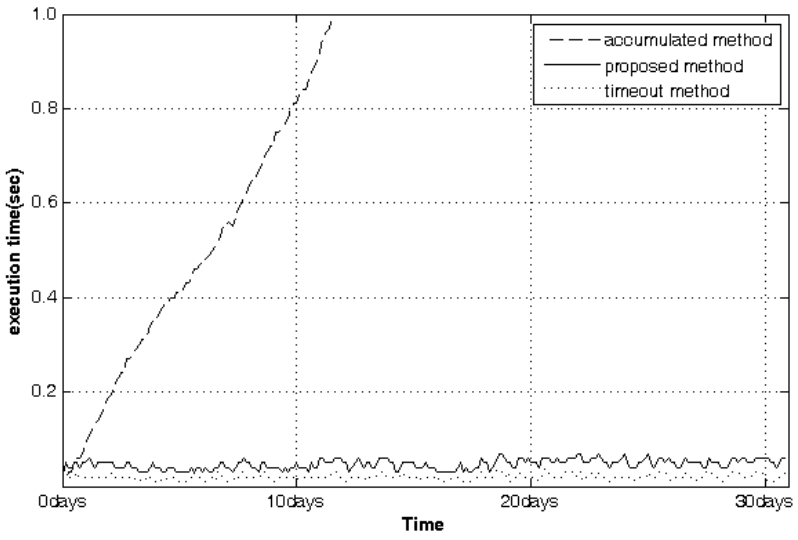

그림 10. Comparison of execution time

그림 10 은 세 방법론의 수행 시간을 비교한 것이다. 본 실험에서는 관리 주기 1 분 마다 시그니쳐 생성과 관리를 수행하는 시간을 비교 분석 하였다. 수행 시간은 생성 네트워크의 트래픽의 양과 보유

시그니쳐에 비례한다. 따라서 가장 많은 시그니쳐를 보유한 accumulated method 가 지속적으로 수행시간이 증가하였고, 이와 대조적으로 proposed method 와 timeout method 는 일정 시간을 유지하였다.

실험 결과를 종합해 보면 분석률 측면에서는 관리 방법 없이 생성된 시그니쳐를 계속 저장하는 방법(accumulated method)에 비해 낮은 성능을 보이지만, 그 외의 평가 요소에서는 일정 시간 사용하지 않는 시그니쳐를 삭제하는 방법론(timeout method)과 비슷한 성능을 보인다. 특히, 시그니쳐 평균 LD 에서는 다른 방법론에 비해 매우 좋은 성능을 보이는 것을 확인할 수 있다. 이로써, 제안한 관리 함수가 시그니쳐의 특성을 잘 파악하여, 향후 분석에 활용될 가능성이 있는 시그니쳐를 잘 유지시키는 것을 알수있다.

6. 결론 및 향후 연구

인터넷 응용 트래픽을 정확하게 분석하기 위해 다양한 분석 방법론이 제안되었다. 하지만, 실제 네트워크 관리 정책에 적용하기에는 아직 한계점과 문제점을 가지고 있다. 기존 문제점을 보완하기 위하여 헤더 시그니쳐가 제안되었지만, 급격히 늘어나는 시그니쳐의 개수는 분석 시스템의 오버헤드로 작용하였다.

본 논문에서는 생성된 헤더 시그니쳐 중에서 분석에 활용도가 높은 시그니쳐를 유지시키는 관리 방법을 제안하였다. 이를 위해 시그니쳐 특징을 파악하여 관리 요소를 정의하고 이를 사용하여 관리함수를 제안하였다. 실험을 통해 관리 방법을 적용하지 않은 경우와 단순히 timeout 값을 적용한 방법보다 더 좋은 분석 성능을 보이는 것을 확인 할 수 있었다. 특정 네트워크의 트래픽을 대상으로 실험하여 보편적인 결과라 단정지을 순 없지만, 본 논문에서 정의한 관리 요소를 활용하여 시그니쳐를 관리한다면 좀 더 효과적으로 헤더 시그니쳐를 활용할 수 있을 것이다.

향후 연구로 다양한 관리 요소를 정의하고 이를 이용한 관리 함수를 제안하겠다. 또한, 객관적 평가를 위해 다양한 네트워크의 트래픽을 대상으로 실험 할 계획이다.

3. 참고 문헌

[1]Myung-Sup Kim, Young J.Won, James Won-Ki Hong, “Application-Level Traffic Monitoring and an Analysis on IP Networks”, ETRI Journal Vol. 27, No.1, February 2005.

[2]S. Sen, J. Wang, “Analyzing peer-to-peer traffic across large networks,” Internet Measurement Conference (IMC), Proc. Of the 2nd ACM SIGCOMM Workshop on Internet measurement, pp 137-150, 2002.

[3]Sung-Ho Yoon, Jun-Sang Park, Myung-Sup Kim, “Signature maintenance for Internet application traffic identification using header signatures,” Proc. of the 4th IEEE/IFIP International Workshop of the Management of the Future Internet (ManFI 2012), Hwaii, USA, Apr. 16, 2012.

[4]Internet Assigned Numbers Authority list, http://www.iana.org/assignments /port-numbers.

[5]A. Moore, K. Papagiannaki, “Toward the Accurate Identification of Network Applications,” Proc. PAM 2005, Boston, USA, 2005.

[6]T. Karagiannis, A.Broido, M. Faloutsos, and kc claffy.Transport layer identification of P2P traffic. In ACM/SIGCOMM IMC, 2004.

[7]M. Baldi, A. Baldini, N. Cascarano, and F. Risso, “Service-based traffic classification: Principles and validation”, Proc. of the IEEE 2009 Sarnoff Symposium, Princeton, NJ, USA, Mar. 2009sk .

윤 성 호

2009년 고려대학교 컴퓨터 정보학과 졸업

2011년 고려대학교 컴퓨터 정보학과 석사

2011년 ~ 현재 고려대학교 컴퓨터 정보학과 박사과정

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

박 준 상

2008년 고려대학교 컴퓨터 정보학과 졸업

2010년 고려대학교 컴퓨터 정보학과 석사

2010년 ~ 현재 고려대학교 컴퓨터 정보학과 박사과정

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 트래픽 분류

김 명 섭

1998년 포항공과대학교 전자 계산학과 졸업

2000년 포항공과대학교 컴퓨터 공학과 석사

2004년 포항공과대학교 컴퓨터 공학과 박사

2006년 Post-Doc. Dept. of ECE, Univ. of Toronto, Canada

2006년 ~ 현재 고려대학교 컴퓨터정보학과 부교수

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 멀티미디어 네트워크